**A framework for Mobile Crowd Sensing and Computer based Systems**

**Introduction:** As the development of technologies like Internet of Things and Big Data continues mankind has enterred an age with unprecedented information. The integration of sensing and embedded everyday computing devices at the edge of the Internet will result in the evolution of an embedded Internet or the Internet of Things. Typical IoT devices include physical items tagged/embedded with sensors (e.g. chemical containers with temperature sensors), scissors with IC-tags, and smart meters to remotely monitor energy consumption. An emerging category of edge devices that we believe will result in the evolution of Internet of Things are consumer centric mobile sensing and computing de-

vices, which are connected to the Internet. These include smartphones (iPhone, Google Nexus), music

players (iPods), sensor embedded gaming systems (Wii, XboX Kinect), and in-vehicle sensing devices

(GPS, OBD-II). They have become extremely popular recently and are potentially important sources

of sensor data. They are typically equipped with various sensing faculty and wireless capabilities

that allow them to produce data and upload the data to the Internet. Different from the ”typical” everyday IoT objects (e.g., coffee machines) that traditionally lack computing capabilities, these mobile devices have a variety of sensing, computing and communication faculty. They can serve either as a bridge to other everyday objects, or generate information about the environment themselves. We believe they will drive a plethora of IoT applications that elaborate our knowledge of the physical world. These applications can be broadly classified into two categories, personal and community sensing, based on the type of phenomena being monitored. In personal sensing applications, the phenomena is pertaining to an individual. For example, the monitoring of movement patterns (e.g. running, walking, exercising) of an individual for personal record-keeping or healthcare reasons. Another example of personal sensing is one that monitors the transportation modes of an individual to determine his or her carbon footprint. On the other hand, community sensing pertains to the monitoring of large-scale phenomena that cannot be easily measured by a single individual. For example, intelligent transportation systems may require traffic congestion monitoring and air pollution level monitoring. These phenomena can be measured accurately only when many individuals provide speed and air quality information from their daily commutes, which are then aggregated spatio-temporally to determine congestion and pollution levels in cities. Community sensing is also popularly called par-

ticipatory sensing [1] or opportunistic sensing [2]. Participatory sensing requires the active involve-

ment of individuals to contribute sensor data (e.g. taking a picture, reporting a road closure) related

to a large-scale phenomena. Whereas opportunistic sensing is more autonomous and user involvement

is minimal (e.g. continuous location sampling without the explicit action from the user). We take

the position that community sensing spans a wide spectrum of user involvement, with participatory

sensing and opportunistic sensing at the two ends. We therefore coin the term mobile crowd sensing

(MCS) to refer to a broad range of community sensing paradigms 1 .

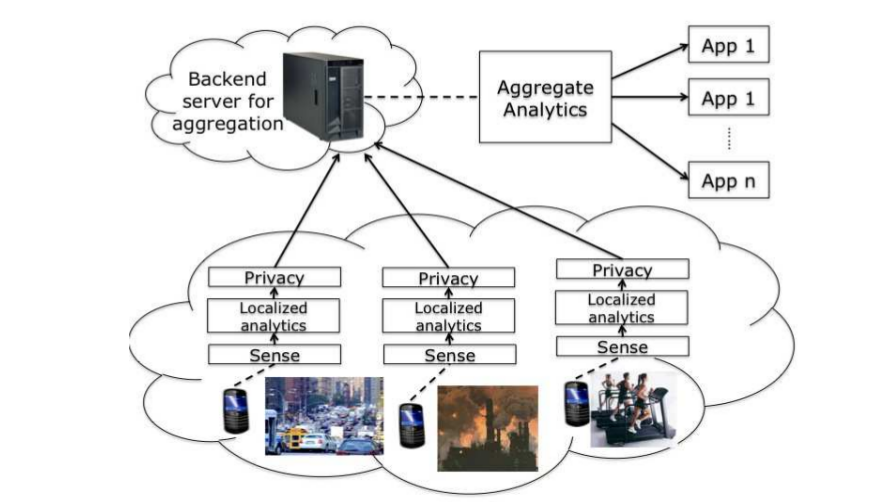
**Crowdsensing**, sometimes referred to as **mobile crowdsensing**, is a technique where a large group of individuals having mobile devices capable of sensing and computing (such as [smartphones](https://en.wikipedia.org/wiki/Smartphones), [tablet computers](https://en.wikipedia.org/wiki/Tablet_computers), [wearables](https://en.wikipedia.org/wiki/Wearables)) collectively share data and extract information to measure, map, analyze, estimate or infer (predict) any processes of common interest. In short, this means [crowdsourcing](https://en.wikipedia.org/wiki/Crowdsourcing) of sensor data from mobile devices. Devices equipped with various sensors have become ubiquitous. Most smartphones can sense ambient light, noise (through the microphone), location (through the [GPS](https://en.wikipedia.org/wiki/Global_Positioning_System)), movement (through the [accelerometer](https://en.wikipedia.org/wiki/Accelerometer)), and more. These sensors can collect vast quantities of data that are useful in a variety of ways. For example, GPS and accelerometer data can be used to locate potholes in cities, and microphones can be used with GPS to map [noise pollution](https://en.wikipedia.org/wiki/Noise_pollution). Mobile crowdsensing belongs to three main types: environmental (such as monitoring pollution), infrastructure (such as locating potholes), and social (such as tracking exercise data within a community).

In environmental MCS applications, the phenomena are those of the natural environment. Examples include measuring pollution levels in a city, water levels in creeks, and monitoring wildlife habitats. Such applications enable the mapping of various large scale environmental phenomena by involving the common man. An example prototype deployment for pollution monitoring is Common Sense [3]. Common Sense uses specialized handheld air quality sensing devices that communicate with mobile phones (using Bluetooth) to measure various air pollutants (e.g. CO 2 , NO x ). These devices when deployed across a large population, collectively measure the air quality of a community or a large area. Similarly, one can utilize microphones on mobile phones to monitor noise levels in communities. Another example is CreekWatch developed by IBM Almaden Research Center. It monitors water levels and quality in creeks by aggregating reports from individuals, such as pictures taken at various locations along the creek, or text messages about the amount of trash. Such information can be used by the water control boards to track pollution levels in water resources. Infrastructure applications involve the measurement of large scale phenomena related to public infrastructure. Examples include measuring traffic congestion, road conditions, parking availability, outages of public works (e.g. malfunctioning fire hydrants, broken traffic lights), and real-time transit tracking. Early MCS deployments measured traffic congestion levels in cities, examples of which include MIT’s CarTel [4] and Microsoft Research’s Nericell [5]. CarTel utilizes specialized devices installed in cars to measure the location and speed of cars and transmit the measured values using public WiFi hotspots to a central server. This central server can then be queried to provide information such as least delay routes or traffic hotspots. On the other hand, Nericell utilizes individuals’ mobile phones to not only determine average speed or traffic delays, but also detect honking levels (especially in countries like India where honking is common) and potholes on roads. Another example is ParkNet [6], an application that detects available parking spots in cities using ultrasonic sensing devices installed on cars combined with smart phones. Finally, the third category is social applications, where individuals share sensed information

amongst themselves. As an example, individuals can share their exercise data (e.g. how much time

one exercises in a day) and compare their exercise levels with the rest of the community. They can use

this comparison to help improve their daily exercise routines. Example deployments include BikeNet [7] and DietSense [8]. In BikeNet, individuals measure location and bike route quality (e.g. CO 2 content on route, bumpiness of ride) and aggregate the data to obtain “most” bikeable routes. In DietSense, individuals take pictures of what they eat and share it within a community to compare their eating habits. A typical use case for this is for a community of diabetics to watch what other diabetics eat and control their diet or provide suggestions to others. To summarize, the functioning of typical MCS applications is illustrated in Figure 2, which depicts a number of research challenges as functional components.



Based on the type of involvement from the users, mobile crowdsensing can be classified into two types:

* [Participatory crowdsensing](https://en.wikipedia.org/w/index.php?title=Participatory_Crowdsensing&action=edit&redlink=1), where the users voluntarily participate in contributing information.[[2]](https://en.wikipedia.org/wiki/Crowdsensing" \l "cite_note-2)
* [Opportunistic crowdsensing](https://en.wikipedia.org/w/index.php?title=Opportunistic_Crowdsensing&action=edit&redlink=1), where the data is sensed, collected and shared automatically without user intervention and in some cases, even without the user's explicit knowledge.

Taking advantage of the ubiquitous presence of powerful mobile computing devices (especially smartphones) in the recent years, it has become an appealing method to businesses that wish to collect data without making large-scale investments. Numerous technology companies use this technique to offer services based on the big data collected, some of the most notable examples being [Facebook](https://en.wikipedia.org/wiki/Facebook), [Google](https://en.wikipedia.org/wiki/Google) and [Uber](https://en.wikipedia.org/wiki/Uber_(company)).

**Process of Crowd Sensing:** Mobile crowdsensing occurs in three stages: data collection, data storage and data upload.

Data collection draws on sensors available through the [Internet of things](https://en.wikipedia.org/wiki/Internet_of_things).[[4]](https://en.wikipedia.org/wiki/Crowdsensing" \l "cite_note-MIST-4) There are three main strategies for collecting this data:[[5]](https://en.wikipedia.org/wiki/Crowdsensing" \l "cite_note-:0-5)

* The user of a device collects data manually. This can include taking pictures or using smartphone applications.
* The user can manually control data collection, but some data can be collected automatically, such as when a user opens an application.
* Data sensing is triggered by a particular context that has been predefined (e.g., a device begins to collect data when the user is in a particular place at a particular time).

The data collection phase can also involve a process called [deduplication](https://en.wikipedia.org/wiki/Data_deduplication), which involves removing redundant information from a data set in order to lower costs and improve user experience.[[5]](https://en.wikipedia.org/wiki/Crowdsensing" \l "cite_note-:0-5) The deduplication process filters and compresses the data that has been collected before it gets uploaded.

**Architecture: A** typical MCS application has two application specific components, one on the device (for sensor data collection and propagation) and the second in the backend (or cloud) for the analysis of the sensor data to drive the MCS application. This architecture is depicted in Figure 3. We refer to this as application silos because each application is built ground-up and independent from each other. There is no common component even though each application faces a number of common challenges in data collection, resource allocation and energy conservation. Such an architecture hinders the development

and deployment of MCS applications in several ways. First, it is hard to program an application. To write a new application, the developer has to address challenges in energy, privacy, and data quality in an ad hoc manner, reinventing the wheel all the time. Further, he may need to develop different variants of local analytics if he wants to run the application on heterogeneous devices using different OSes. Second, this approach is inefficient. Applications performing sensing and processing activities independently without understanding the consequences on each other will result in low efficiency on an already resource constrained platform. Thereis a high likelihood of duplicating sensing and processing across multiple applications. For example, traffic sensing, air and noise pollution all require location information, but these applications would each do its own sampling without reusing the same data samples. Further, there is no collaboration or coordination across devices. Devices may not all be needed (e.g. traffic sensing in a given location) especially when the device population is dense. Finally, the current architecture is not scalable. Only a small number of applications can be accommodated on each device (e.g. limitations imposed by the device operating system, human capacity to keep track of a large number of applications). Also, the data gathered from societal-scale sensing may overwhelm network and backend server capacities, thus making the current architecture non-scalable.

Various sensors such as GPS, accelerometer, microphone and camera are available on mobile devices. The OS allows applications to access the sensors and extract raw sensing data from them. However, depending on the nature of the raw data and the needs of applications, the physical readings from sensors may not be suitable for the direct consumption of applications. Many times, some local analytics performing certain primitive processing of the raw data on the device are needed. They produce intermediate results which are sent to the backend for further processing and consumption. For example, in a pothole detection [5] application, a local analytic computes spikes from 3-axis acceleration sensor data to determine potential potholes. The motivation of such local analytics are two-

fold. First, the kind of processing performed leads to appropriately summarized data, thus consuming lesser energy and bandwidth than transmitting the raw sensor readings. This is a well-known tradeoff in conventional mote-class sensor networks: using computation to save energy/bandwidth. Second, it reduces the amount of processing that the backend has to perform. Further, if the mobile devices in a societal scale deployment transmit raw sensor data, the backend can easily be overwhelmed. Finally, some applications are delay sensitive and transmitting raw sensor data on intermittently connected channels can be time consuming as compared to that of sending processed sensor data.

Compared to traditional mote-class sensor networks, mobile crowdsensing has a number of unique characteristics that bring both new opportunities and problems. First, today’s mobile devices have significantly more computing, communication and storage resources than mote-class sensors, and they are usually equipped with multimodality sensing capabilities. These will enable many applications that require resources and sensing modalities beyond current mote-class sensors possess. Second, millions of mobile devices are already “deployed in the field”: people carry these devices wherever they go and whatever they do. By leveraging these devices, we could potentially build large scale sensing applications efficiently (cost and time). For example, instead of installing road-side cameras and loop detectors, we can collect traffic data and detect congestion levels using smartphones carried by drivers. Such solutions reduce the cost of deployment of specialized sensing infrastructure. The dynamic conditions of the set of mobile devices and the need for data reuse across different applications in MCS are also quite different from those of traditional sensor networks. In MCS, the population of mobile devices, the type of sensor data each can produce, and the quality in terms of accuracy, latency, confidence can change all the time due to device mobility, variations in their energy levels and communication channels, and device owners’ preferences. Identifying the right set of devices that can produce the desired data and instructing them to sense with proper parameters to ensure desired quality is a complex problem. In traditional sensor networks, the population and the data they can produce are mostly known apriori, thus controlling the data quality is much easier. The same sensor data have been used for different purposes in many existing MCS applications. For example, the accelerometer readings have found usage in transportation mode identification, pothole detection, human activity pattern extraction. To efficiently support multiple concurrent applications, it is critical to identify common data needs and support the reuse of sensor data across applications. In contrast, a conventional sensor network is typically intended for a single application and reuse for vastly different purposes is rarely needed. Because devices are owned and carried by individual users, humans are usually involved in the loop. On one hand, the intelligence and mobility of humans can be leveraged to help applications

collect higher quality or semantically complex data that may otherwise require sophisticated hardware

and software. For example, humans can easily identify available street parking spots and report with pictures or text messages, whereas an ultrasound based scanning system not only requires special hardware but also sophisticated processing algorithms to ensure the reliability of data. On the other hand, humans naturally have privacy concerns and personal preferences that are not necessarily aligned with the end goals of the MCS applications. The user may not want to share sensor data that contains or reveals private and sensitive information, such as their current location. Another important implication for human involvement are incentives. Participating individuals (devices) may incur energy, monetary costs or even explicit efforts on the owner of the device for sensing, processing and communicating of desired data. Unless there are strong enough incentives, the owners may not be willing to contribute their resources. For MCS applications to succeed, there have to be appropriate incentive mechanisms to recruit, engage, and retain human participants. Elaboration on incentive mechanisms and other people-oriented tools are beyond the scope of this paper, since our focus is on system challenges.

**CROWD COMPUTING:** It is an overarching term which defines the myriad tools that enable idea sharing, non-hierarchical decision making and the full utilization of the world’s massive "[cognitive surplus](https://en.wikipedia.org/wiki/Cognitive_surplus)”-the ability of the world’s population to collaborate on large, sometimes global projects. Crowd computing brings together the strengths of [crowdsourcing](https://en.wikipedia.org/wiki/Crowdsourcing), [automation](https://en.wikipedia.org/wiki/Automation), [distributed computing](https://en.wikipedia.org/wiki/Distributed_computing), and [machine learning](https://en.wikipedia.org/wiki/Machine_learning). Crowd Computing is defined as harnessing the power of people out in the web to do tasks that are hard for individual users or computers to do alone. Like cloud computing, crowd computing offers elastic, on-demand human resources that can drive new applications and new ways of thinking about technology. Crowd computing offers a harmonious amalgamation of both cloud computing and crowdsourcing. It combines human intelligence (the crowd) with artificial intelligence (the cloud) in order to produce quality results at unprecedented speed. Scientists and historians are already utilizing this process to complete time-consuming research, and many businesses are beginning to realize its potential for cutting costs and increasing productivity. Crowd computing very well may be on its way to changing the way humans live and operate in our society by using artificial intelligence in combination with the human mind. crowd computing will complement the cloud as one of two burgeoning infrastructures that will enable the world to become more ‘collectively intelligent’ . Four key process and technology innovations are the core of crowdcomputing:

* Microtasking. Work is broken into small components that are easier to complete by the crowd.
* Automation. Machines complete repetitive work, leaving judgement work to humans
* Hybrid Crowd: Higher value work and a greater volume of work can be completed when specialists, crowd workers and machines work together.
* AI: Machine learning creates a cascade of knowledge that enables more and more automation and continuously optimizes cost and quality.

Businesses and society in general increasingly rely on the combined intelligence, knowledge, bandwidth and life experiences of the ‘crowd to improve processes, make decisions, identify solutions to complex problems and monitor changes in consumer taste. Companies like [Amazon](https://en.wikipedia.org/wiki/Amazon.com) and [Google](https://en.wikipedia.org/wiki/Google) saw early-on the potential for crowd computing. In 1995, Amazon created [Mechanical Turk](https://en.wikipedia.org/wiki/Amazon_Mechanical_Turk) to deal with its internal problem of sorting its massive inventory. The platform organizes people from around the globe to ‘work efficiently as a giant machine.”Google uses a [captcha](https://en.wikipedia.org/wiki/Captcha) to help digitize books. Major sites like [Facebook](https://en.wikipedia.org/wiki/Facebook) and [Twitter](https://en.wikipedia.org/wiki/Twitter) rely on the crowd to power the translation that spreads their service around the globe.

**Bluetooth as a Used Case of Crowd Sensing:**

We used Bluetooth low energy (BLE) tagging as an alternative method. When low-cost BLE tags are set in advertisement mode, they can be detected by smartphones. In this paper, we design an architecture for sensing the crowds by requiring a large population carrying relatively cheap off-the-shelf BLE proximity tags, and considerably fewer participants to run scanning application on their smartphones to collect data. In recent years, Bluetooth Low Energy (BLE) has emerged as a new wireless personal area network (WPAN) technology. There are millions of BLE enabled accessories shipped in 2013 and nearly 2.6 billion expected to be shipped by 2016 [1]. The technology can be embedded into a very small form factor as as in Figure 2. BLE tags are very cheap, lasting years with small battery, and actively advertising its presents to nearby readers. Most importantly, almost all the latest versions of mobile platforms have been working to provide native support for BLE, and are already everywhere in the market.

**BLE:**

**Offline Helper:**

**Aim:** Is to build an application that helps user get direction information from a source to destination through Bluetooth.

**Situation:** Two or more persons have to start the service of the application which denotes that they either want to help others or they need help. The one who needs help does not have internet connection but the one who is willing to help must have internet connection. The one who needs help enters the source destination and traveling mode and sends this information as a broadcast message over the network on click the ‘ask for help’ button. Others who have started the service have this running in the background and get a notification upon receiving the broadcast message. They can then click on the notification and the direction guidance will be fetched from the internet which they can send via Bluetooth to the one who needed it. The user id is saved when a broadcast message is received and this reply with the direction information is sent as a unicast message. This part is done with the help of a bridgefy sdk.

**About Bridgefy:** Bridgefy Sdk is for android and iOS and also allows message passing between the two OS. Bridgefy allows messages to “hop” multiple times without device jail breaking required, while preserving battery life, security, and allowing for Android - iOS communication. Messages are shared through autonomous mesh networks created by mobile devices by using their Bluetooth and Wi-Fi antennas to allow information to travel large distances without requiring an Internet connection. It allows direct message passing between two devices using Bluetooth upto a distance of 330 ft. It uses mesh network that is messages and requests can be sent to others through hop ie using devices in between. Each device provides an extra 330 feet. It also allows messages to be broadcasted all throughout the network. An API key needs to be generated and the first time it requires Internet because it verifies if the key is registered or not. The SDK have to be added to the project dependencies. Then we can create and instance and start it. It has methods for sending messages, receiving requests which need to be overridden.

**About Google Directions API:** Google provides an API that returns the direction details of two places as a json or xml response. The source, destination and traveling mode(optional by default driving) needs to be passed to a link. An http request is made with that link which returns the response in the xml format. The xml response is then parsed and displayed as a text in the scrollable textview area named answers.

**Setup:** i) A registered API key is taken from bridgefy.me website which lets us use the bridgefy SDK in the background.

ii) A google direction API key is taken which lets us have the direction information through a Http request.

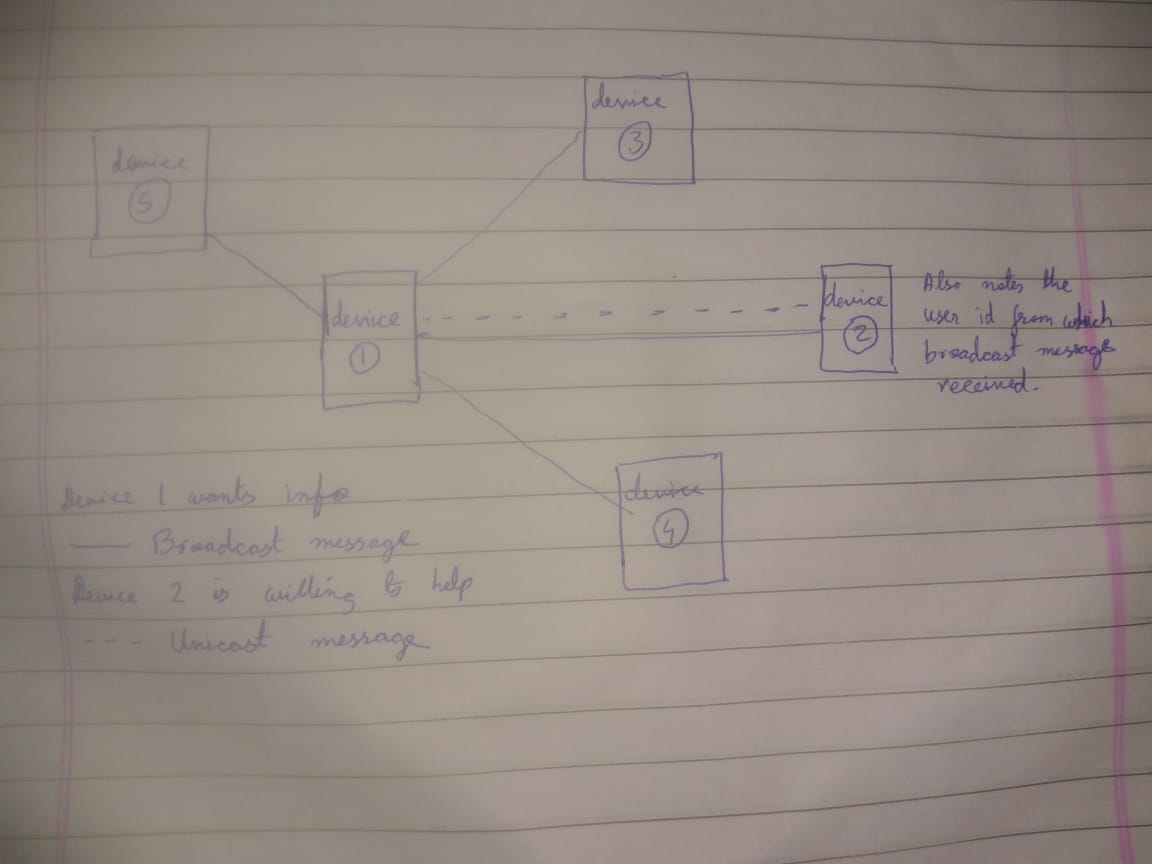
iii) Two devices willing to send messages and get direction information between source and destination.

iv) Both the devices should allow the application access to Bluetooth, Bluetooth Admin and Fine Location and Internet.

v) On the first time installation the devices must have internet as the application verifies is the API key from bridgefy SDK is registered or not.

vi) The two devices should be within 330m of one another because Bluetooth message passing is not possible over more than that distance.

**Implementation Diagram:**



**Working :** Initially both the users (one who wants help-user1 and the one who wants to help- user2) has to start a service. The service button will start a service in the background and bluetooth will be turned on. Then user1 will have to choose if he wants to send a text query or wants to receive direction information from a particular source to a particular destination via a particular travel mode. He will then click on ask for help which will broadcast the query message across the network and all the devices which have their services on will receive the notification. A particular user say user2 wants to help. He will click on the notification and the application will be opened. He will have two options. If he knows the particular answer he can directly write the reply and sent the message which will be sent as an unicast reply or he can fetch the answer from the google direction API and the message will be sent automatically. User1 without having Internet will get the direction from the Internet which is fetched by another user and is passed over bluetooth to user1.